REMARKS/ARGUMENTS

Claims 1-34 are pending in the present application. Claim 34 was canceled; claims 1, 12, 29 and 32 were amended; and claim 35 was added. Support for these amendments may be found in the claims as originally filed and in paragraph [0052] of the specification. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 102, Anticipation

The examiner has rejected 1-34 under 35 U.S.C. § 102 as being anticipated by Christfort et al. (U.S. Publication No. 2002/0078168), hereinafter "*Christfort*". This rejection is respectfully traversed.

The examiner states:

> Consider Claim 29, Christfort disclosed A computer readable media storing instructions to be executed by a processor of a computer system (Christfort, [0298]), said processor of the computer system executing an integrated development environment (IDE) for generating code for executing in a client-server environment (Christfort, [0080], Christfort discloses an IDE for executing code), said instructions defining an extensible mechanism for executing said code on a server that (Christfort, [0081], [0084]), when deployed on said computer system, adapts said IDE (Christfort, [0080]-[0084]) to: process an input object identifying code for executing on one of a plurality of servers (Christfort, [0022], Christfort disclosed on identifying several types of input objects which can be used for coding for the application) said processing using a view list of at least one input object element (Christfort, [0022], Christfort discloses on view several input object codes which are presented to the user), each input object element processing a type of code identified by the input object to output a deployable object (Christfort, [0022], Christfort discloses on how the application code is selected by an object on the interface, and the application code may be executed in a response to a request for a service from an end user); process the deployable object using a server list of at least one server element to determine the one of the plurality of servers for executing the code (Christfort, [0062], Christfort does show the server list for example containing host servers on a portal page), each server element enabling the deployable object to execute on a particular server and outputting a launchable object (Christfort, [0094]-[0095], Christfort disclosed on how objects/created applications are launched via the system); and process the launchable object using a launcher list of at least one client element to determine a client for launching the code on the one of the plurality of servers (Christford, [0093], Christfort indicates on how portal-to-go

XML document or application program containing the code generates the output, and how the output is launched by the system).

Office Action dated December 24, 2008, pages 2-3.

A prior art reference anticipates the claimed invention under 35 U.S.C. §102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). The *Christfort* reference cited by the examiner does not anticipate the present invention as recited in the claims, because *Christfort* fails to teach each and every element of the claims.

Independent claim 29, which is representative of independent claims 1, 12, and 32 with regard to similarly recited subject matter, reads as follows:

> 29. A computer readable media storing instructions to be executed by a processor of a computer system, said processor of the computer system executing an integrated development environment (IDE) for generating code for executing in a client-server environment, said instructions defining an extensible mechanism for executing said code on a server that, when deployed on said computer system, adapts said IDE for handling new code types by:
>
> processing an input object identifying code for executing on one of a plurality of servers, said processing using a view list of at least one input object element, each input object element processing a type of code identified by the input object to output a deployable object;
>
> processing the deployable object using a server list of at least one server element to determine the one of the plurality of servers for executing the code, each server element enabling the deployable object to execute on a particular server and outputting a launchable object; and
>
> processing the launchable object using a launcher list of at least one client element to determine a client for launching the code on the one of the plurality of servers.

Christfort does not teach instructions defining an extensible mechanism for executing said code on a server that, when deployed on the computer system, adapts the integrated development environment for handling new code types. The examiner points to the following sections of *Christfort* as teaching these features:

> **[0081]** According to one embodiment, a developer uses a browser running on a client to log on to a website through which the developer can access the tools necessary to develop an application without any special client-side SDK software. The website may be referred to as a development website, or as an "SDK website" when, for example, the website provides, or hosts, an online software development kit (SDK).

> **[0082]** As used herein, a "developer" is synonymous with both a "service provider" and a "user" (since the developer makes use of the online development website). Also, a "user" is distinguished from an end-user or a customer that uses the service provided by the developer.
> **[0083]** The provider of the application development tools and host of the online development website may be referred to as the development provider, SDK provider, or simply as the host. The development provider typically uses one or more servers to support the development website and the tools provided thereby.
> **[0084]** Access to the development website may be controlled by requiring the user to have a name and password. After the user has gained access to the website, the user may be presented with a list of mobile applications (also referred to herein as services), that the developer or user has previously created. The user may also have access to other services, such as samples provided by the development website provider. The development website may present the user with several options, such as to add a new service, modify an existing service, or delete an existing service. For example, if the user selects the" add service" option, the user is presented with a user interface that allows the user to add a service.

Christfort, paragraphs [0081]-[0084].

The cited paragraphs of *Christfort* above disclose that a user may access necessary tools to develop an application by logging on to a website. The website is referred to as a development website or an SDK website when the website provides an online software development kit. When the user gains access to the development website, the website may present the user with a list of mobile applications (services) previously created by the user. The website may also present the user with several options, such as to add a new service, modify an existing service, or delete an existing service.

While *Christfort* provides a method for developing applications, *Christfort* does not teach defining an *extensible mechanism* that *adapts the integrated development environment to handle new code types*. *Christfort* teaches creating and modifying online applications using an online software development kit. For instance, paragraph [0080], reproduced below, further describes the *Christfort* process:

> **[0080]** Host server 110 also includes an online software development kit 116 that services and application that developers may use to create, edit, test, deploy, and otherwise manage applications or modules associated with the hosting service. According to one embodiment, an application developer uses a browser to log into an online development website associated with the hosting service to access online software development kit 116. After logging in, online software development kit 116 provides a user interface for display on the developer's web browser. The user interface presents the user with various options, such as creating a new application, modifying an existing application, testing an

> application, or deleting an existing application. For example, to create a hosted application, the developer enters or modifies program code using the user interface via the developer's browser and then saves the code at host server 110 under a module or application name, or identifier. If the developer wishes to establish a shared hosted application, the developer logs into the online development website and provides a URL of the application. The hosted and shared hosted applications may then be made immediately available for use. For example, end users may log into the service provider via a website to access a list of available services.

As described above, the *Christfort* reference teaches logging onto an online development website associated with a hosting service to access an online software development kit, which provides a user interface in which a user may create, modify, test, or delete applications through the user interface. While *Christfort* teaches the creation and modification of on-line applications using an on-line SDK for later deployment to servers, claim 29 of the present invention recites a process for modifying a software development kit itself using the extensible mechanism, which allows adapting to new code types. The presently claimed invention enables the IDE to adapt to new code types prior to installation on the server that is identified in order for the server to be able to run the new code. *Christfort* does not mention anything about having an IDE or SDK "adapt" to the new instructions using an extensible mechanism. No such modification of the IDE/SDK itself is performed in *Christfort*. Consequently, *Christfort* does not teach instructions defining an extensible mechanism for executing said code on a server that, when deployed on the computer system, adapts the integrated development environment to handle new code types as recited in the presently claimed invention.

Christfort also does not teach processing an input object identifying code for executing on one of a plurality of servers, said processing using a view list of at least one input object element, each input object element processing a type of code identified by the input object to output a deployable object. The examiner points to the following sections of *Christfort* as teaching these features:

> **[0022]** Techniques are provided for developing applications online. According to one aspect, applications for services are developed online. A request is received from a user running a browser on a client to develop a new application. In response to the request, a first electronic document, such as a web page, is provided to the client for display on the user's browser of an application development interface. The interface may include several types of objects, including an edit field for typing in code for an application. The user submits the application code by selecting an object on the interface, such as a submit button.

> The application code is stored at a server that is remote from the client and from where the application code may be executed in response to a request for service from an end user. Similarly, the interface may be used to retrieve and edit existing application code, or to test, deploy, or delete existing application code.

Christfort, paragraph [0022].

Paragraph [0022] of *Christfort* discloses identifying several types of input objects which can be used for coding for the application. A view of several input object codes are presented to the user. Application code is submitted by selection of an object on the interface, and the application code may be executed in a response to a request for a service from an end user. However, as previously mentioned, *Christfort* is merely directed to the creation and modification of on-line applications using an on-line software development kit. *Christfort* only teaches the selection of input types to develop a client/server application, rather than an extensible mechanism in an IDE/SDK that allows support of additional program code types. Thus, *Christfort* merely teaches the execution of an application built by the SDK subsequent to a service request, rather than an input object element that processes a type of code identified by the input object to output a deployable object for the type of code.

Christfort further does not teach processing the deployable object using a server list of at least one server element to determine the one of the plurality of servers for executing the code, each server element enabling the deployable object to execute on a particular server and outputting a launchable object. The examiner points to the following sections of *Christfort* as teaching these features:

> **[0062]** Host server 110 may be implemented on one or more servers at an intermediary, such as a hosting service provider, also known as a host provider or simply as a host. The function of the host is to install and maintain applications, such as on host server 110, that are developed by either the host provider or other application developers. The applications are typically part of a service, such as a web site, a paging service, or a telecommunications service. The host may also provide "partial" or "shared" hosting of applications in which the applications are stored on servers associated with the application developer or service provider, but the applications may be accessed via the host. Partial or shared hosting of applications is distinguished from portal applications that are stored on servers associated with the application developer or service provider but which are not accessed via the host. End users access the services offered by other parties and companies via the host by interacting with the hosted and partially hosted applications.

Christfort, paragraph [0062].

> **[0094]** As soon as a service has been created and/or revised, end users or customers that can connect to the network on which the server resides (e.g., the Internet) can access the service. The process by which the service is accessed may vary based on the type of end user. For example, a desktop computer can connect to the Internet through a dial-up line, a DSL connection, a cable modem, an ISDN connection or many other available methods. WAP phones may connect to the Internet over a wireless connection using a synchronous protocol, such as through a WAP-to- HTTP gateway, or using an asynchronous protocol, such as the simple mail transfer protocol (SMTP) or the short message service (SMS) protocol.
> **[0095]** In general, the end user logs in to the development website and is presented with a list of available services. The end user may select the service that was just created and/or modified. In response to the selection of a service, the application associated with the service is obtained. The application may be associated with a portal-to-go XML document or another document written in a suitable markup language. For example, with hosted applications, the XML may be obtained by retrieving the portal-to-go XML that was saved at a server associated with the intermediary (e.g., the host). For portal applications, a request may be sent to the URL that was specified by the developer for the service. The web server that manages that URL may invoke the portal application and send the portal-to-go XML associated with the portal application back to the server associated with the intermediary. Once the portal-to-go XML for the service is obtained for either a hosted or portal application, the server associated with the intermediary uses the portal-to-go XML to provide the selected service to the client associated with the end user.

Christfort, paragraphs [0094]-[0095].

Paragraph [0062] of *Christfort* discloses a host provider whose function is to install and maintain applications that are part of a service. End users may access the services via the host. However, *Christfort* does not teach processing the deployable object to determine the one of the plurality of servers for executing the code. No processing of a deployable object is disclosed as occurring in *Christfort*, as the target servers in *Christfort* is already known. The process disclosed in *Christfort* goes from a user creating or editing an application directly to saving the application on a server as a new or existing service, as explicitly disclosed in the following section below:

> **[0091]** When **the user finishes inputting the code for a new application or editing the code for an existing application,** the user can save the code shown in the interface. The "save" code function may be an object, such as a button, included in the interface. By clicking on the object, **the code is submitted, or sent, to a server** remote from the client, such as a server associated with the development provider. If a new application is being created, the user specifies a service name and the code is saved in association with the service name. After saving the new code, the new service name will be displayed in a list of existing

> services shown when users and end users log into the development website. If an existing application is being modified, the user may choose to save the code under the existing service name or under a new service name.

Christfort, paragraph [0091] (emphasis added).

Thus, while *Christfort* supports a plurality of servers for a hosting environment, *Christfort* does not support the explicitly recited feature of processing the deployable object to determine the one of the plurality of servers for executing the code as recited in the presently claimed invention. The *Christfort* process does not mention a processing step using a deployable object that determines which one of a plurality of servers that will execute the code. Instead, *Christfort* teaches a simpler server side application environment where the user has already chosen the server to target prior to editing and saving the application. Thus, *Christfort* does not disclose a deployable object being processed to determine the one of the plurality of servers for executing the code. In addition, *Christfort* does not disclose a server element that enables the deployable object to execute on a particular server, nor that the server element outputs launchable object. *Christfort* also makes no mention of an input object that outputs a deployable object for the type of code, and processing of the deployable object creates an output of a launchable object.

Christfort also does not teach processing the launchable object using a launcher list of at least one client element to determine a client for launching the code on the one of the plurality of servers. The examiner points to the following section of *Christfort* as teaching these features:

> **[0093]** In another embodiment, to create a shared hosted application, the user writes either a portal-to-go XML document or an application program that generates a portal-to-go XML document as output. The terms "partially hosted application" or the "shared hosted application" may be used to refer either to the XML document or the application that generates an XML document as output. The shared hosted application may be saved, for example, at the application developer's own website. The user then associates a URL with the shared hosted application using, for example, an HTTP listener/web server that services the application developer's web site. The shared hosted application is then added as a "service" by logging into the development website, or the SDK website, and providing the name of the service and the URL associated with the shared hosted application.

Christfort, paragraph [0093].

Paragraph [0093] discloses how a shared hosted application may be created by a user by writing either to a portal-to-go XML document or application program containing the code to

generate a portal-to-go XML document as output, and how the portal-to-go XML document output is launched by the system. However, the presently recited invention recites processing a launchable object. *Christfort* is merely teaching creating server side applications that can run on any type of client. Further the user is required to perform the manual step disclosed in paragraph [0095]:

> **[0095]** In general, the end user logs in to the development website and is presented with a list of available services. The end user may select the service that was just created and/or modified. In response to the selection of a service, the application associated with the service is obtained. The application may be associated with a portal-to-go XML document or another document written in a suitable markup language.

Christfort, paragraph [0095].

A launchable object is defined in paragraph [0041] of the present specification as "an output, such as a Java code object, XML document etc., that contains information on how to access code to be run (i.e. a specific resource) on a particular server, and information about how to route access to the resource, (e.g. firewalls and/or the type of client application that can be used to access the web resource) if the access information is not sufficiently clear. For example, a URL may be constructed and provided to direct access to the resource when a Web browser is intended to consume the object. Thus, while *Christfort* supports any client launching the service application, *Christfort* makes no mention of pre-configuring a launcher at a client element to support launchable objects. Consequently, *Christfort* does not teach processing the launchable object using a launcher list of at least one client element to determine a client for launching the code on the one of the plurality of servers.

Since claims 2-11, 13-27, 30-31 and 35 depend from claims 1, 12, and 29, respectively, the same distinctions between *Christfort* and the claimed invention in claims 1, 12, and 29 also apply to these dependent claims. Furthermore, these dependent claims also include additional features not found in the *Christfort* reference. For example, claims 2 and 13 recite processing the input object to identify the code for executing on the one of the plurality of servers includes using a view list of at least one input element for processing a type of code identified by the input object, processing the generated code includes using a server list of at least one server element for determining the one of the plurality of servers, and identifying the one of the plurality of client applications includes using a launcher list of at least one client element for

launching the one of the plurality of client applications. While the examiner refers again to paragraphs [0022] and [0093] of *Christfort* (reproduced above) as teaching these features, paragraph [0022] merely discloses several types of input objects which can be used for coding for an application. However, paragraph [0022] does not mention anything about processing an input object for the purpose of subsequently identifying code to be executed by a server. In addition, paragraph [0022] also fails to teach using a view list of at least one input element for processing a type of code identified by the input object as in the presently claimed invention, as *Christfort* only discloses the selection of input types to develop a client/server application. *Christfort* fails to teach using input type elements in manner recited in the presently claimed invention. Furthermore, paragraph [0093] of *Christfort* merely discloses how a portal-to-go XML document or application program containing the code generates a portal-to-go XML document output, and how the portal-to-go XML document output is launched by the system. In contrast, claim 3 recites processing the generated code includes using a server list of at least one server element for determining the one of the plurality of servers. *Christfort* does not teach and is not concerned with selecting the correct server to use for the type of code. *Christfort* discloses that a user may select a service manually, but does not support processing code to determine which server to use for the code type.

Claims 3, 4, 5, 14, 16, and 18 recite that the view list, server list and launcher list that are extensible to accommodate additional respective elements. *Christfort* does not teach any extensibility feature of a view list, server list, or launcher list as in the presently claimed invention.

Claims 6, 7, 21, and 22 recite that processing the input object comprises analyzing the input object to determine an input object element for processing the input object and processing the input object using the determined input object element and processing user input to determine the input object element. The cited sections of *Christfort* teach identifying several types of input objects that can be used for coding for the application. However, the SDK provider in *Christfort* does not process an input object for the purpose of subsequently identifying code to be executed by a server.

Claims 8, 9, 15, and 16 recite that processing the generated code comprises analyzing a server element for enabling a deployable object and processing the deployable object using the determined server element. The cited sections of *Christfort* teach creating code that will work on

any client device. In contrast, claims 8, 9, 15, and 16 recite processing a deployment object that enables a server to enable a server side application to run on that server. *Christfort* does not teach any analysis of server side elements or teach wherein such analysis is used for enabling a deployable object.

Claims 10, 11, 17, and 18 recite that identifying the one of the plurality of client applications comprises analyzing a launchable object to determine a client element for processing the launchable object and processing the launchable object using the determined client element. The cited sections of *Christfort* teach creating server side applications that can run on any type of client. In contrast, claims 10, 11, 17, and 18 recite identifying a client application by analyzing a launchable object to determine the client element to process the launchable object. *Christfort* does not teach any analysis of a launchable to determine a client application for processing the launchable object.

Claim 35 recites perform a compatibility test of the input object, deployable object, and launchable object prior to processing the input object and display a result of the compatibility test to a user. *Christfort* does not teach such a compatibility test.

Therefore, the rejection of claims 1-34 under 35 U.S.C. § 102 has been overcome.

Furthermore, *Christfort* does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. *Christfort* actually teaches away from the presently claimed invention because it teaches the creation and modification of on-line applications using an on-line software development kit for later deployment to servers as opposed to modifying a software development kit itself using an extensible mechanism that allows for adapting to new code types as in the presently claimed invention. Absent the examiner pointing out some teaching or incentive to implement *Christfort* and an extensible mechanism for executing code on a server that adapts an integrated development environment to support additional code types, one of ordinary skill in the art would not be led to modify *Christfort* to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion, or incentive to modify *Christfort* in this manner, the presently claimed invention can be reached only through an improper use of hindsight using the applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

II. Conclusion

It is respectfully urged that the subject application is patentable over the cited reference and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: March 24, 2009

Respectfully submitted,

/Cathrine K. Kinslow/

Cathrine K. Kinslow
Reg. No. 51,886
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants